

# UTILITY PATENT APPLICATION TRANSMITTAL

06-30-1998

Submit an original and a duplicate for fee processing  
(Only for new nonprovisional applications under 37 CFR 1.53(b))

Jc523 U.S. PTO  
09/114231

06/30/98

## ADDRESS TO:

Assistant Commissioner for Patents  
Box Patent Application  
Washington, D.C. 20231

Attorney Docket No. 84505  
First Named Inventor Ilan Gabriel Caron  
Express Mail No. EM597736253US  
Total Pages 57

## APPLICATION ELEMENTS

1. ☒ Transmittal Form with Fee
2. ☒ Specification (including claims and abstract) [Total Pages 48]
3. ☒ Formal Drawings [Total Sheets 8]
4. ☒ Unexecuted Combined Declaration and Power of Attorney [Total Pages 2]
  - a. ☐ Newly executed
  - b. ☐ Copy from prior application  
[Note Box 5 below]
    - i. ☐ Deletion of Inventor(s) Signed statement attached deleting inventor(s) named in the prior application
5. ☐ Incorporation by Reference: The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied under Box 4b is considered as being part of the disclosure of the accompanying application and is hereby incorporated by reference therein.
6. ☐ Microfiche Computer Program
7. ☐ Nucleotide and/or Amino Acid Sequence Submission
  - a. ☐ Computer Readable Copy
  - b. ☐ Paper Copy
  - c. ☐ Statement verifying above copies

## ACCOMPANYING APPLICATION PARTS

8. ☐ Assignment Papers
9. ☐ Power of Attorney
10. ☐ English Translation Document (if applicable)
11. ☐ Information Disclosure Statement (IDS)
  - ☐ PTO-1449 Form
  - ☐ Copies of IDS Citations
12. ☐ Preliminary Amendment
13. ☒ Return Receipt Postcard (Should be specifically itemized)
14. ☐ Small Entity Statement(s)
  - ☐ Enclosed
  - ☐ Statement filed in prior application; status still proper and desired
15. ☐ Certified Copy of Priority Document(s)
16. ☒ Other: Cover Sheet [1 page]

17. If a **CONTINUING APPLICATION**, check appropriate box and supply the requisite information:  
☐ Continuation ☐ Divisional ☐ Continuation-in-part of prior application Serial No.

## APPLICATION FEES

BASIC FEE				\$790.00
CLAIMS	NUMBER FILED	NUMBER EXTRA	RATE	
Total Claims	66 -20=	46	x \$22.00	\$1012.00
Independent Claims	9 - 3=	6	x \$ 82.00	\$ 492.00
<input type="checkbox"/> Multiple Dependent Claims(s) if applicable			+\$270.00	\$
Total of above calculations =				\$
Reduction by 50% for filing by small entity =				\$( )
<input type="checkbox"/> Assignment fee if applicable			+\$40.00	\$
TOTAL =				\$2294.00

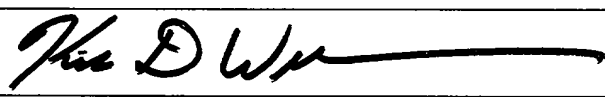
## UTILITY PATENT APPLICATION TRANSMITTAL

Attorney Docket No. 84505

18. ☐ Please charge my Deposit Account No. 12-1216 in the amount of \$ .
19. ☒ A check in the amount of \$2,294.00 is enclosed.
20. The Commissioner is hereby authorized to credit overpayments or charge any additional fees of the following types to Deposit Account No. 12-1216:
- a. ☒ Fees required under 37 CFR 1.16.
  - b. ☒ Fees required under 37 CFR 1.17.
  - c. ☐ Fees required under 37 CFR 1.18.
21. ☒ The Commissioner is hereby generally authorized under 37 CFR 1.136(a)(3) to treat any future reply in this or any related application filed pursuant to 37 CFR 1.53 requiring an extension of time as incorporating a request therefor, and the Commissioner is hereby specifically authorized to charge Deposit Account No. 12-1216 for any fee that may be due in connection with such a request for an extension of time.

## 22. CORRESPONDENCE ADDRESS

Kirk D. Williams, Registration No. P42,229  
Leydig, Voit & Mayer, Ltd.  
Two Prudential Plaza, Suite 4900  
180 North Stetson  
Chicago, Illinois 60601-6780  
Telephone: (312) 616-5600  
Facsimile: (312) 616-5700

Name	Kirk D. Williams
Signature	
Date	June 30, 1998

## Certificate of Mailing Under 37 C.F.R. 1.10

I hereby certify that this Utility Patent Application Transmittal and all accompanying documents are being deposited with the United States Postal Service "Express Mail Post Office To Addressee" Service under 37 C.F.R. 1.10 on the date indicated below and is addressed to: Assistant Commissioner for Patents, Box Patent Application, Washington, D.C. 20231.

		June 30, 1998
Name of Person Signing	Signature	Date

**S P E C I F I C A T I O N**

To all whom it may concern:

Be it known that I, **ILAN GABRIEL CARON**, a citizen of the **United States**, residing at **1265 23RD AVENUE EAST, SEATTLE, WASHINGTON 98112** has invented a certain new and useful **METHOD AND APPARATUS FOR CREATING, SENDING, AND USING SELF-DESCRIPTIVE OBJECTS AS MESSAGES OVER A MESSAGE QUEUING NETWORK**, of which the following is a specification.

**METHOD AND APPARATUS FOR  
CREATING, SENDING, AND USING SELF-DESCRIPTIVE OBJECTS  
AS MESSAGES OVER A MESSAGE QUEUING NETWORK**

5                   **FIELD OF THE INVENTION**

                  This invention relates to computer programming and  
networking, and more particularly to an automated  
method and computer apparatus for sending and using  
self-descriptive objects as messages over a message  
10   queuing network.

**BACKGROUND OF THE INVENTION**

                  Users and developers of networked applications and  
systems desire reliable, faster and easier to use  
methods of communicating information between source and  
15   destination computer applications and operating  
environments. Traditional messaging techniques require  
each application to know the specific serialized format  
of a message, or require communication between the  
operating environments of the sender and receiver to  
20   provide information or meta-data so that the receiver  
can interpret the message. Computer users and  
applications developers are desirous of new methods and  
computer apparatus for communicating messages which

decrease the amount of configuration and runtime overhead involved.

Most distributed computing applications today use synchronous communication technologies, such as remote  
5 procedure calls. Such synchronous communications require a sender of a request to wait for a response from the receiver of the request before it can proceed and perform other tasks. The time that the sender must wait depends on the time it takes for the receiver to  
10 process the request and return a response. Synchronous communication mechanisms also require the sender and the receiver to be operating simultaneously.

In contrast, using asynchronous communications, senders make requests to receivers and can move on to  
15 perform other tasks immediately. If a response is expected back from the receiver, it is up to the original sender to decide when it will actually look for and process the response. Most importantly, there is no guarantee that receivers will process requests  
20 within any particular period of time. In fact, with asynchronous communications, there are no requirements that receivers be running nor even the communications infrastructure be available in order for a sender to initiate a request.

Message queuing systems implement asynchronous communications by enabling applications to send messages to and receive messages from other applications. These applications may be running on the same machine or on separate machines connected by a network. When an application receives a request message, it processes the request by reading the contents of the message formatted in a known pattern and acting accordingly. If required, the receiving application can send a response message back to the original requestor.

Many applications are now using message queuing networks for the enhanced communication delivery reliability between networked computer systems provided by sending messages asynchronously across a message queuing enterprise network. However, these messages are simply received as type-less buffers of raw data that are passed between applications. In some instances, these messages have additional signaling information attached that describe how the message should be sent by the underlying sub-system. However, the messages do not provide any semantic information that enables the message recipient to interpret the meaning of the message contents. To communicate, the

source and destination applications rely either on private message content encoding schemes or prior arrangements between the applications to only send messages of a certain type.

5

**SUMMARY OF THE INVENTION**

According to the invention, an automated method and apparatus are provided for creating, sending, and using self-descriptive objects as messages between applications, and additionally sending these message objects over a message queuing network. Required meta-information is included with these self-descriptive messages making them self-contained and requiring no external components to interpret them. Using the present invention, networked applications can communicate arbitrary objects in a standard way with no prior agreement as to the nature and semantics of message contents. In this manner, applications are more robust and can readily adapt to changes to message contents without having to update the format or structure of the message, or to update the application to interpret the encoded body of a new message format.

In one embodiment of the present invention, messages are sent as serialized dictionary objects over

a message queuing network. The dictionary represents an abstract data type defined in terms of four fundamental operations that can be performed on it, namely: add, remove, lookup, and enumerate. These  
5 operations correspond to methods invoked to perform the desired operation. As implied by the method names, add() adds a specified element to the dictionary; remove() removes a specified element in the dictionary; lookup() finds a specified element in the dictionary;  
10 and enumerate() returns one element from the dictionary, allowing the retrieval of all elements from the dictionary.

The dictionary elements, in an embodiment of the present invention, are in the form of a triplet  
15 comprised of a Name, Type and Value. The Name represents a string identifier; the Type specifies the type of element which could be as simple as a constant or integer, or be a more complex (and very rich) type such as an Excel spreadsheet or even another serialized  
20 data dictionary; and the Value specifies a current value or state of the element. The previously described triplet merely illustrates a very generalized abstract data element. Various other dictionary data

elements could be employed in keeping with the present invention.

To enable the dictionary object to be sent across a network, the dictionary object is able to serialize and deserialize itself using two more of its methods. 5 The save() method causes the dictionary object to serialize itself to the body of a message, and the load() method loads into the object a previously serialized dictionary object located in the body of a 10 received message.

In accordance with the present invention, a sender application creates a persistent dictionary object, and populates the object with the desired contents of the message. The sender application then requests the 15 dictionary object to save or serialize itself into the body of a message queuing message (or the dictionary object could be serialized into a buffer which is copied or moved into the body of a message queuing message prior to sending the message). The message 20 queuing system forwards the message containing the serialized object to the destination queue.

Upon receipt from the destination queue, the receiving message queuing system looks at the received message, and determines that it contains a dictionary

object in the body of the message. The destination message queuing system then instantiates and loads the message object with the data dictionary, and passes the object to the recipient application.

5       The recipient application then uses the dictionary object in any manner it chooses. In one embodiment of a recipient application, the recipient application enumerates the elements of the data dictionary and takes appropriate programming action for each element  
10       according to its type. For example, a received Excel spreadsheet in a dictionary element could cause the application to start an Excel application and to forward the value of the element (i.e., the Excel spreadsheet) to the Excel application. Other  
15       dictionary elements might contain a single integer, or records containing multiple fields which would be processed accordingly by the recipient application. Thus, the present invention provides a generalized and robust messaging mechanism whereby the sending and  
20       receiving applications no longer rely on a previous agreed to protocol format or a specialized serialization scheme.

**BRIEF DESCRIPTION OF THE DRAWINGS**

The appended claims set forth the features of the present invention with particularity. The invention, together with its advantages and as previously  
5 described, may be better understood from the following detailed description taken in conjunction with the accompanying drawings of which:

FIG. 1A is a block diagram of an exemplary operating environment in which the invention may be  
10 implemented, including a computer network comprising computer systems for sending and using self-descriptive objects as messages over a message queuing network in accordance with the invention;

FIG. 2A is a block diagram illustrating the  
15 transmission of messages in a message queuing environment;

FIG. 2B is a block diagram illustrating sites within a message queuing environment;

FIG. 2C is a block diagram illustrating connected  
20 networks within a message queuing environment;

FIG. 3A is a block diagram illustrating the an embodiment of a persistent dictionary object with its interfaces and methods;

FIG. 3B is a block diagram illustrating an exemplary format of the serialized dictionary object;

FIG. 4A is a flow diagram illustrating the steps performed by an application to send a message object;

5        FIG. 4B is a flow diagram illustrating the steps performed by an application using a received message object;

FIG. 5A is a flow diagram illustrating the steps performed by a MSMQ server to serialize and send a  
10    message object; and

FIG. 5B is a flow diagram illustrating the steps taken by a MSMQ server in response to receiving a serialized message object.

#### **DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT**

15        FIG. 1 and the following discussion are intended to provide a brief, general description of a suitable computing environment in which the invention may be implemented. Although not required, the invention will be described in the general context of  
20    computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform

particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including

5 hand-held devices, multiprocessor systems, microprocessor-based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where  
10 tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

15 With reference to FIG. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a conventional personal computer 20, including a processing unit 21, a system memory 22, and a system bus 23 that couples various  
20 system components including the system memory to the processing unit 21. The system bus 23 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The

system memory includes read only memory (ROM) 24 and random access memory (RAM) 25. A basic input/output system 26 (BIOS) containing the basic routines that helps to transfer information between elements within the personal computer 20, such as during start-up, is stored in ROM 24. In one embodiment of the present invention on a server computer 20 with a remote client computer 49, commands are stored in system memory 22 and are executed by processing unit 21 for creating, sending, and using self-descriptive objects as messages over a message queuing network in accordance with the invention. The personal computer 20 further includes a hard disk drive 27 for reading from and writing to a hard disk, not shown, a magnetic disk drive 28 for reading from or writing to a removable magnetic disk 29, and an optical disk drive 30 for reading from or writing to a removable optical disk 31 such as a CD ROM or other optical media. The hard disk drive 27, magnetic disk drive 28, and optical disk drive 30 are connected to the system bus 23 by a hard disk drive interface 32, a magnetic disk drive interface 33, and an optical drive interface 34, respectively. The drives and their associated computer-readable media provide nonvolatile storage of computer readable

instructions, data structures, program modules and other data for the personal computer 20. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 29 and a removable optical disk 31, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROM), and the like, may also be used in the exemplary operating environment.

A number of program modules may be stored on the hard disk, magnetic disk 29, optical disk 31, ROM 24 or RAM 25, including an operating system 35, one or more application programs 36, other program modules 37, and program data 38. A user may enter commands and information into the personal computer 20 through input devices such as a keyboard 40 and pointing device 42. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 21 through a serial port interface 46 that is coupled to the system

bus, but may be collected by other interfaces, such as a parallel port, game port or a universal serial bus (USB). A monitor 47 or other type of display device is also connected to the system bus 23 via an interface, such as a video adapter 48. In addition to the monitor, personal computers typically include other peripheral output devices (not shown), such as speakers and printers.

The personal computer 20 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 49.

The remote computer 49 may be another personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the personal computer 20, although only a memory storage device 50 has been illustrated in FIG. 1. The logical connections depicted in FIG. 1 include a local area network (LAN) 51 and a wide area network (WAN) 52. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

When used in a LAN networking environment, the personal computer 20 is connected to the local network

51 through a network interface or adapter 53. When  
used in a WAN networking environment, the personal  
computer 20 typically includes a modem 54 or other  
means for establishing communications over the wide  
5 area network 52, such as the Internet. The modem 54,  
which may be internal or external, is connected to the  
system bus 23 via the serial port interface 46. In a  
networked environment, program modules depicted  
relative to the personal computer 20, or portions  
10 thereof, may be stored in the remote memory storage  
device. It will be appreciated that the network  
connections shown are exemplary and other means of  
establishing a communications link between the  
computers may be used.

15 The present invention provides for sending  
self-descriptive message objects as messages between  
two or more applications, and operates in any computing  
environment that supports data objects, whether on a  
standalone computer or preferably in a networked  
20 environment. Using self-descriptive objects as  
messages, a recipient no longer relies on a convention  
or a special-coding serialization scheme. The  
recipient application can simply extract a data element  
from the received object in a standard, well-known way,

discover the element's logical type, and take appropriate programmatic action.

The present invention is described in the context of a Microsoft Message Queue Server (MSMQ) network and using Microsoft Component Object Model (COM) objects in order to illustrate one embodiment of the invention. The present invention is not so limited, as the teachings disclosed herein provide for the present invention to be used in other messaging systems and communications networks, as well as using other forms of objects and self-descriptive structures.

A brief introduction of message queuing is provided below. A more detailed explanation of MSMQ is described in "Microsoft Message Queue Server (MSMQ)," MSDN Library - April 1998, Microsoft Corporation, and is hereby incorporated by reference. And a detailed explanation of COM is described in "COM and ActiveX Object Services," MSDN Library - April 1998, Microsoft Corporation, and is hereby incorporated by reference.

MSMQ implements asynchronous communications by enabling applications to send messages to, and receive messages from, other applications. These applications may be running on the same machine or on separate

machines connected by a network. MSMQ messages can contain data in any format that is understood by both the sender and the receiver. When an application receives a request message, it processes the request by  
5 reading the contents of the message and acting accordingly. If required, the receiving application can send a response message back to the original requestor.

While in transit between senders and receivers, MSMQ keeps messages in holding areas called queues,  
10 hence the name message queuing. MSMQ queues protect messages from being lost in transit and provide a place for receivers to look for messages when they are ready. Applications make requests by sending messages to queues associated with the intended receiver. If  
15 senders expect responses in return, they must include the name of a response queue (that the sender must create in advance) in all requests that they make to the receiver.

Turning now to FIG. 2A, shown is a block diagram  
20 illustrating the basics of the transportation of a message 75 from message queuing machine 1 (computer 80) to machine 2 (computer 90) over a transport network 85 supporting such network transport protocols as TCP/IP or IPX. The message 75 contains self-descriptive

objects and/or self-descriptive data elements in  
accordance with the present invention. Each computer  
80 and 90 performs both server and client operations  
for transferring messages 75 between their respective  
5 message queues.

A message queuing enterprise network can span many  
locations and operate on top of different transport  
network protocols. The topology of the message queuing  
enterprise network can be described in terms of  
10 (1) physical location and (2) communication protocol  
connectivity. The term "site" describes an aspect of  
the enterprise network based on a physical location.  
In contrast, a "connected network" describes an aspect  
of the message queuing enterprise network according to  
15 communication protocol connectivity.

An enterprise network is a collection of sites  
connected through slow/expensive network connections.  
A site, is a physical collection of machines, where  
communication between two machines is cheap and fast.  
20 These two computers are typically located in the same  
physical location, although not required. The concept  
of a site is integral to the message routing algorithm  
employed by the message queuing system. In order to  
route messages throughout the message queuing

enterprise network, a message queuing computer must be able to locate the destination message queue. A subset of computers within the message queuing network are also directory servers ("DS servers") which maintain message queuing information, including information to enable routing of messages such as sites, connected networks, and names of DS servers within the message queuing network.

A MSMQ network is a collection of addresses "speaking" several communication protocols and are connected by physical communication links. A connected network is a collection of addresses, where every two addresses can communicate directly (i.e., the underlying communication network provides the connection if all its components are on-line). Inside a connected network, communication delay and cost may vary. The physical communication lines and the traffic overhead define the communication delay and cost. Two addresses in a connected network may be connected by a fast, cheap line, for example, if their machines are in the same site or by a slow expensive line if their machines are in different sites. Two machines belong to the same connected network if they support the same protocol, and can have a direct session on that

protocol. A machine can support more than one  
connected network on a specific protocol if it supports  
more than one address which belong to different  
connected networks on a specific protocol. A connected  
5 network does not consist of more than one protocol.

These concepts are further illustrated in  
FIGs. 2B-C, shown in block diagrams illustrating an  
enterprise network 200. As illustrated in FIG 2B,  
shown are three sites: site A (201), site B (202),  
10 site C (203), connected by network lines 212, 213, and  
223. As previously described herein, sites are a  
grouping of computers within a message queuing network  
grouped together for the purposes of routing. One  
distinction that can be made between sites in a typical  
15 message queuing network is that sites are connected to  
relatively slow, expensive lines. Computers within a  
site are typically connected by fast, cheap lines such  
as those computers residing on a single Ethernet. For  
example, site A (201) contains a plurality of message  
20 queuing computers 230, 231 connected by fast networking  
lines 234. These computers can also perform additional  
message queuing functionality. For example, computer  
231 might be a DS server. In addition, computer 232

might be a remote access server (RAS) with software to respond to client requests packets.

Turning now to FIG. 2C, illustrated is an enterprise network 200 showing sites A-C (201-203) and connected networks 261-264. As previously described herein, each connected network within a message queuing network represents those machines which can directly communicate with each other using a single networking protocol, such as TCP/IP or IPX. As shown in FIG. 2C, computers 270-272, 280-282 and 290-291 support TCP/IP protocol, and computers 283, 290, 294 support IPX protocol. A computer can use more than one protocol as represented by computer 290, or support more than one network interface for the same protocol as represented by computers 270 and 280. In addition, a computer can be connected to more than one connected network. For example, computers 270 and 280 belong to two connected IP networks 261 and 262; and computer 290 belongs to two connected networks 261 and 264 supporting IP and IPX protocols. It is also possible for a connected network to span all sites, such as illustrated by connected network 261 spanning sites A-C (201-203).

In one embodiment of the present invention, messages are sent as serialized dictionary objects over

a message queuing network. The dictionary represents an abstract data type defined in terms of four fundamental operations that can be performed on it, namely: add, remove, lookup, and enumerate; with the  
5 addition of two operations to serialize and unserialize the persistent dictionary object to enable the dictionary object to be sent across a network.

Turning now to FIG. 3A, shown is a block diagram illustrating persistent dictionary object 300  
10 comprising an IDictionary interface 310 and an IPersistDict interface 320. The dictionary object 300 contains a data structure and methods that when invoked, perform operations on the internal data structure. The operations performed on the data  
15 elements correspond to methods invoked to perform the desired operation. As implied by the method names, add() 301 adds a specified element to the dictionary; remove() 302 removes a specified element in the dictionary; lookup() 303 finds a specified element in  
20 the dictionary; and enumerate() 304 provides a mechanism for obtaining the next element from the dictionary given a position in the dictionary. To enable the dictionary object to be sent across a network, the save() method 321 causes the dictionary

object to serialize itself to a specified target location (i.e., the message body) and the load() method 322 loads a serialized dictionary object.

The dictionary elements, in an embodiment of the  
5 present invention, are in the form of a triplet  
comprised of a Name, Type and Value. The Name  
represents a string identifier; the Type specifies the  
type of element which could be as simple as a constant  
or integer, or be a more complex (and very rich) type  
10 such as an Excel spreadsheet or even a serialized data  
dictionary; and the Value specifies a current value or  
state of the element. In an embodiment, the type field  
contains an agreed upon indicator specifying the type  
of element (e.g., 1 is an integer, 2 is a string, 3 is  
15 an object, etc.). In another embodiment, the type  
mechanism is extended to provide a standard way for  
receivers to learn about type indicators that the  
receiver does not recognize such as by querying the  
sending application, the message queuing network, or  
20 some other local or remote process.

For example, a record of data such as an address  
book entry could be sent as a persistent dictionary  
object, with the address book entries being defined in  
terms of two dictionary elements. The first dictionary

element having a Name of "Entry Name", being of Type "string", and having a Value of "USPTO"; with the second dictionary element having a Name of "City", being of Type "string", and having a Value of

5 "Washington D.C.". Using Visual Basic and dimensioning *d* as a New PersistentDictionary, the elements could be added to *d* using the statements:

*d*.Add("Entry Name","USPTO"), and

*d*.Add("City"," Washington D.C.").

10 Then, the elements could be extracted from *d* by the following references: *d*("Entry Name") and *d*("City").

Using the previously described triplet as a data element merely illustrates a very generalized abstract data element. Various other dictionary data elements  
15 could be employed in keeping with the present invention. In addition, late binding techniques could be used to make each named element in the data dictionary a data member of the object. Using this technique, elements of the dictionary could be  
20 referenced directly. For example, a data element *msword\_document* in a dictionary *d* could be referenced as *d.msword\_document* as opposed to *d("msword\_document")*.

Turning now to FIG 3B, illustrated is a serialized dictionary object 360. The first field, CElements 370, contains the number of elements in the serialized dictionary object 360, which is followed by each of the dictionary elements. As shown, the first dictionary element 380 comprises the triplet of the Name 381, Type 382 and Value 383. A dictionary object can contain a plurality of dictionary elements as indicated by element field 399.

FIGS. 4A, 5A, 5B, and 4B illustrate the steps performed by a sending application, the sending MSMQ server, the receiving MSMQ server, and the recipient application, respectively, in sending a message object from a sending application to a recipient application over a MSMQ network in one embodiment. In other embodiments, certain of these described functions could be performed by the application instead of the message queuing network and vice versa. For example, the serialization and deserialization of the persistent dictionary object could be performed by the sending and recipient applications (or by other intermediate protocol layers, or by other processes). In this example, the message queuing network would not necessarily need to know that it was transporting a

self-descriptive message. Moreover, self-descriptive messages (e.g., persistent dictionary objects) could be transported using other network technologies and protocols, in addition to, or in place of the message queuing network described herein.

First, turning to FIG. 4A, illustrated are the steps performed by a Microsoft Visual Basic application preparing and sending a message object containing an Excel spreadsheet across a MSMQ network. First, a MSMQ queue  $q$ , an Excel spreadsheet  $x1$ , and a MSMQ message  $m$  are dimensioned in steps 405-415. Next, the body of the message  $m$  is set to the Excel spreadsheet  $x1$  in step 420. Finally, in step 425, the MSMQ message  $m$  is sent via queue  $q$ .

Next, turning to FIG. 5A, the sending MSMQ server continues in response to the request to send the message object by the sending application in step 420 (FIG. 4A). First, in step 505, the message object is checked to see if it supports data persistence (such as being a COM object). If it does not support data persistence, then the object is not sent in one embodiment and processing ends with step 545. In other embodiments, it would be possible to add additional functionality based on the teachings disclosed herein

to incorporate serialization and unserialization of arbitrary objects.

Otherwise, if the message object supports persistence as determined in step 505, then the  
5 required size of a buffer is determined and allocated in step 510 to accommodate the serialized message object. Next, in step 515, the persistent storage type supported by the message object is determined. If the  
10 message object supports streams, then processing flows to steps 520-525 wherein the message object writes itself to the buffer, and the message type is set to a "streamed object". Otherwise, the message object supports storage (the other storage type for a COM object) and processing continues with steps 530-535  
15 wherein a storage pointing to the message buffer is created, the object saves itself to the storage (i.e., the message buffer), and the message type is set to a "stored object". Finally, in step 540, the MSMQ message body is set to the contents of the buffer and  
20 the MSMQ server forwards the message to the destination queue.

When such a message object is received at a receiving MSMQ server queue and the message has been determined to contain an object by querying the message

itself using a method of the message, the message is processed according to the flow diagram of FIG. 5B. In step 555, the object message type is evaluated and if it is of a "streamed object" type, then processing  
5 continues with steps 560-565 wherein the received message object creates a stream which is initialized by the message buffer memory, and a class identifier (CLSID) is obtained from the stream. Otherwise, the object message is of a "storage object" type, and steps  
10 570-575 are performed wherein the received message object creates a storage which is initialized by the message buffer memory, and a class identifier (CLSID) is obtained from the storage.

Next, in step 580, the OLE interface  
15 CoCreateInstance is used to instantiate the message object (i.e., the persistent dictionary object). Then, the load method 322 (FIG. 3A) of the instantiated object is invoked to load the serialized data (from the appropriate initialized storage or stream that was  
20 created in step 560 or 570) in step 585. Finally, in step 590, the receiving MSMQ server returns the message object (i.e., the instantiated and loaded dictionary object) to the recipient application in step 590.

The recipient application then uses the received self-contained message object as described herein with reference to the flow diagram of FIG. 4B. First, in step 455, a MSMQ queue *q*, a MSMQ message *m*, and a persistent data dictionary *d* are dimensioned. Next, in 5 step 460, *m* is set to the message received from the sender application via the MSMQ network as explained herein with reference to FIGs. 4A, 5A and 5B. Having obtained the message *m* containing the self-descriptive 10 object, the recipient application processes the message however it desires.

The remaining steps 465-499 illustrate one embodiment of such processing. First, if the body of the received message is not a persistent dictionary as 15 determined in step 465, then the non-persistent data object (e.g., an integer, record, string) is processed by the application. For example, the recipient application could print the address book previously described herein by setting *d* to the message body of a 20 received message containing an address book entry, and then using the statement:

```
print The d("Entry Name") is in d("City")
```

which would print:

```
The USPTO is in Washington D.C.
```

Otherwise, the received message is a persistent dictionary as determined in step 465, and *d* is set to the message body in step 470. Next, while there are  
5 elements remaining in the persistent dictionary *d*, steps 477-495 are performed for each element. In step 477, an element is enumerated from the data dictionary. Next, steps 480-495 are performed which embody a case statement switching upon the `typeof()` the  
10 element (i.e., the type of the persistent dictionary element received in the MSMQ message). For example, if the type of the element is an Excel spreadsheet, then Excel operations are performed. Otherwise, processing continues in the case statement with a generic type  
15 "CaseType" provided for illustrative purposes in steps 490, 495 to signify the diverse and rich types of elements that can be sent across a network in a self-descriptive message using the present invention. This CaseType could be any data type, including an integer,  
20 string, data record, address book entries, or even a persistent dictionary. Many different configurations are also possible, including the recipient application being a CaseType application and processing the received element, or a CaseType application being

invoked by the recipient application or message queuing system to process the received the data element.

In view of the many possible embodiments to which the principles of our invention may be applied, it will  
5 be appreciated that the embodiment described herein with respect to the drawing figures is only illustrative and should not be taken as limiting the scope of the invention. To the contrary, the invention as described herein contemplates all such embodiments  
10 as may come within the scope of the following claims and equivalents thereof.

What is claimed is:

1. In a computer system, a method for sending a data element from a sending application in the computer system to a recipient application in the computer system, the method comprising the steps of: the sending application requesting the computer system to deliver the data element to the recipient application; the computer system adding the data element to an object; the computer system encoding the object containing the data element; the computer system unencoding the object; the computer system extracting the data element from the object; and the recipient application receiving the data element from the computer system.

2. The computer system of claim 1, wherein the sending and recipient applications are on separate computers connected via a network.

3. The method of claim 2, wherein the network comprises a message queuing network.

4. The method of claim 3, wherein the step of encoding the object includes requesting the object to serialize itself.

5. The method of claim 3, wherein the step of unencoding the object includes requesting a new instantiation of the object to load itself.

6. The method of claim 1, wherein the object includes a data structure and a method which performs an operation on the data structure.

7. The method of claim 1, wherein the object is a dictionary object.

8. The method of claim 1, wherein the object supports persistence.

9. The method of claim 8, wherein the step of encoding the object includes requesting the object to serialize itself.

10. The method of claim 1, wherein the data element includes a name, a type, and a value.

11. The method of claim 10, wherein the type of the data element is a constant, an integer, a document, a spreadsheet, a database, an object, or a data structure.

5           12. In a networked computer system, a method for sending a self-descriptive object from a first application to a second application, the first application running on a first computer and the second application running on a second computer, the first and  
10 second computers interconnected via a network, the method comprising the steps of: the first application adding data to the self-descriptive object; the first computer transmitting the self-descriptive object to the second computer; the second computer receiving the  
15 self-descriptive object; and the second application processing the data in the self-descriptive object based on the type of data.

13. The method of claim 12, further comprising the step of the first application requesting the first  
20 computer to send the message to the second computer or to the second application.

14. The method of claim 12, further comprising the step of the second computer passing a pointer to the self-descriptive object to the second application.

15. The method of claim 12, wherein the  
5 self-descriptive object includes a data structure and a method which performs an operation on the data structure.

16. The method of claim 12, wherein the self-descriptive object is a dictionary object.

10 17. The method of claim 12, wherein the self-descriptive object supports persistence.

18. The method of claim 12, wherein the data includes an element comprising a name, a type, and a value.

15 19. The method of claim 18, wherein the type of the data element is a constant, an integer, a document, a spreadsheet, a database, an object, or a data structure.

20. The method of claim 18, wherein the type of data is a spreadsheet; and the second application is a spreadsheet application or the spreadsheet application is invoked to process the data.

5        21. The method of claim 18, wherein the type of data is a document; and the second application is a word processing application or the word processing application is invoked to process the data.

10       22. The method of claim 12, wherein the network comprises a message queuing network.

23. In a message queuing network comprising a first and a second message queuing servers, a method for sending a self-descriptive object from the first message queuing server to the second message queuing  
15    server, the method comprising the steps of: the first message queuing server receiving a request to send the self-descriptive object from a first application; the first message queuing server creating a message which includes the self-descriptive object in its payload;  
20    the first message queuing server transmitting the message over the message queuing network; the second

message queuing server receiving the message; and the  
second message queuing server extracting the  
self-descriptive object from the message.

24. The method of claim 23, further comprising the  
5 step of the second message queuing server passing the  
self-descriptive object to a second application.

25. The method of claim 23, wherein the  
self-descriptive object includes a data structure and a  
method which performs an operation on the data  
10 structure.

26. The method of claim 23, wherein the  
self-descriptive object is a dictionary object.

27. The method of claim 26, wherein the dictionary  
object includes an element comprising a name, a type,  
15 and a value.

28. The method of claim 27, wherein the type of  
the element is a constant, an integer, a document, a  
spreadsheet, a database, an object, or a data  
structure.

29. The method of claim 23, wherein the self-descriptive object supports persistence.

30. The method of claim 23, further comprising the step of the first messaging computer serializing the  
5 self-descriptive object.

31. A message queuing network comprising: a first message queuing server including means to receive a self-descriptive object, means to serialize the self-descriptive object, and means to transmit a  
10 message containing the serialized self-descriptive message; and a second message queuing server including means to receive a message containing the serialized self-descriptive object, and means to unserialize the serialized self-descriptive object.

15 32. The message queuing network of claim 31, wherein the self-descriptive includes a data structure and a method which performs an operation on the data structure.

33. In a message queuing network comprising a  
20 first message queuing machine and a second message

queuing machine, a method for sending a  
self-descriptive dictionary object from a sending  
application to a recipient application, the method  
comprising the steps of: the sending application  
5 passing the dictionary object to the first message  
queuing machine to deliver to the second message  
queuing machine; the first message queuing machine  
invoking a method of the dictionary object to serialize  
the dictionary object; the first message queuing  
10 machine sending the serialized dictionary object in a  
message to the second message queuing machine; the  
second message queuing machine instantiating and  
loading the serialized dictionary object into an  
unserialized dictionary object; and the second message  
15 queuing machine passing the unserialized dictionary  
object to the recipient application.

34. The method of claim 33, further comprising the  
steps of: the sending application adding a data  
element to the dictionary object, the data element  
20 including an identifier, a type, and a value; and the  
recipient application enumerating the data element from  
the dictionary object, and processing the data element  
based on its type.

35. The method of claim 34, wherein the type of the data element is a constant, an integer, a document, a spreadsheet, a database, an object, or a data structure.

5        36. The method of claim 33, wherein the dictionary object includes a data structure and a method which performs an operation on the data structure.

37. A computer-readable medium having computer-executable instructions for performing steps for  
10    sending a data element from a sending application in a computer system to a recipient application in the computer system, the steps comprising: the sending application requesting the computer system to deliver the data element to the recipient application; the  
15    computer system adding the data element to an object; the computer system encoding the object containing the data element; the computer system unencoding the object; the computer system extracting the data element from the object; and the recipient application  
20    receiving the data element from the computer system.

38. The computer-readable medium of claim 37,  
wherein the sending and recipient applications are on  
separate computers connected via a network.

39. The computer-readable medium of claim 38,  
5 wherein the network comprises a message queuing  
network.

40. The computer-readable medium of claim 38,  
wherein the object includes a data structure and a  
method which performs an operation on the data  
10 structure.

41. The computer-readable medium of claim 38,  
wherein the object is a dictionary object.

42. The computer-readable medium of claim 38,  
wherein the object supports persistence.

15 43. The computer-readable medium of claim 38,  
wherein the data element includes a name, a type, and a  
value.

44. The computer-readable medium of claim 43,  
wherein the type of the data element is a constant, an

integer, a document, a spreadsheet, a database, an object, or a data structure.

45. A computer-readable medium having computer-executable instructions for performing steps in a  
5 networked computer system for sending a self-descriptive object from a first application to a second application, the first application running on a first computer and the second application running on a second computer, the first and second computers  
10 interconnected via a network, the steps comprising: the first application adding data to the self-descriptive object; the first computer transmitting the self-descriptive object to the second computer; the second computer receiving the  
15 self-descriptive object; and the second application processing the data in the self-descriptive object based on the type of data.

46. The computer-readable medium of claim 45, having further computer-executable instructions for  
20 performing the step of the first application requesting the first computer to send the message to the second computer or to the second application.

47. The computer-readable medium of claim 45,  
wherein the self-descriptive object includes a data  
structure and a method which performs an operation on  
the data structure.

5        48. The computer-readable medium of claim 45,  
wherein the self-descriptive object is a dictionary  
object.

49. The computer-readable medium of claim 45,  
wherein the self-descriptive object supports  
10 persistence.

50. The computer-readable medium of claim 45,  
wherein the data includes an element comprising a name,  
a type, and a value.

51. The computer-readable medium of claim 50,  
15 wherein the type of the data element is a constant, an  
integer, a document, a spreadsheet, a database, an  
object, or a data structure.

52. The computer-readable medium of claim 50,  
wherein the type of data is a spreadsheet, and having  
20 further computer-executable instructions for performing

the step of the second application invoking a spreadsheet application.

53. The computer-readable medium of claim 50, wherein the type of data is a document, and having  
5 further computer-executable instructions for performing the step of the second application invoking a word processing application.

54. The computer-readable medium of claim 45, wherein the network comprises a message queuing  
10 network.

55. A computer-readable medium having computer-executable instructions for performing steps for sending a self-descriptive object from a first message queuing server to a second message queuing server in a  
15 message queuing network, the steps comprising: the first message queuing server receiving a request to send the self-descriptive object from a first application; the first message queuing server creating a message which includes the self-descriptive object in  
20 its payload; the first message queuing server transmitting the message over the message queuing

network; the second message queuing server receiving the message; and the second message queuing server extracting the self-descriptive object from the message.

5           56. The computer-readable medium of claim 55 having further computer-executable instructions for performing the step of the second message queuing server passing the self-descriptive object to a second application.

10           57. The computer-readable medium of claim 55, wherein the self-descriptive object includes a data structure and a method which performs an operation on the data structure.

15           58. The computer-readable medium of claim 55, wherein the self-descriptive object is a dictionary object.

          59. The computer-readable medium of claim 58, wherein the dictionary object includes an element comprising a name, a type, and a value.

60. The computer-readable medium of claim 59, wherein the type of the element is a constant, an integer, a document, a spreadsheet, a database, an object, or a data structure.

5        61. The computer-readable medium of claim 55, wherein the self-descriptive object supports persistence.

62. The computer-readable medium of claim 55 having further computer-executable instructions for  
10 performing the step of the first messaging computer serializing the self-descriptive object.

63. A computer-readable medium having computer-executable instructions for performing steps for sending a self-descriptive dictionary object from a  
15 sending application to a recipient application in a message queuing network comprising a first message queuing machine and a second message queuing machine, the steps comprising: the sending application passing the dictionary object to the first message queuing  
20 machine to deliver to the second message queuing machine; the first message queuing machine invoking a

method of the dictionary object to serialize the  
dictionary object; the first message queuing machine  
sending the serialized dictionary object in a message  
to the second message queuing machine; the second  
5 message queuing machine instantiating and loading the  
serialized dictionary object into an unserialized  
dictionary object; and the second message queuing  
machine passing the unserialized dictionary object to  
the recipient application.

10 64. The computer-readable medium of claim 63,  
having further computer-executable instructions for  
performing the steps of: the sending application  
adding a data element to the dictionary object, the  
data element including an identifier, a type, and a  
15 value; and the recipient application enumerating the  
data element from the dictionary object, and processing  
the data element based on its type.

65. The computer-readable medium of claim 64,  
wherein the type of the data element is a constant, an  
20 integer, a document, a spreadsheet, a database, an  
object, or a data structure.

66. The computer-readable medium of claim 63,  
wherein the dictionary object includes a data structure  
and a method which performs an operation on the data  
structure.

**ABSTRACT**

An invention for creating, sending, and using self-descriptive objects as messages over a network is disclosed. In an embodiment of the present invention, self-descriptive persistent dictionary objects are  
5 serialized and sent as messages across a message queuing network. The receiving messaging system unserializes the message object, and passes the object to the destination application. The application then  
10 queries or enumerates message elements from the instantiated persistent dictionary, and performs the programmed response. Using these self-descriptive objects as messages, the sending and receiving applications no longer rely on an a priori convention  
15 or a special-coding serialization scheme. Rather, messaging applications can communicate arbitrary objects in a standard way with no prior agreement as to the nature and semantics of message contents.

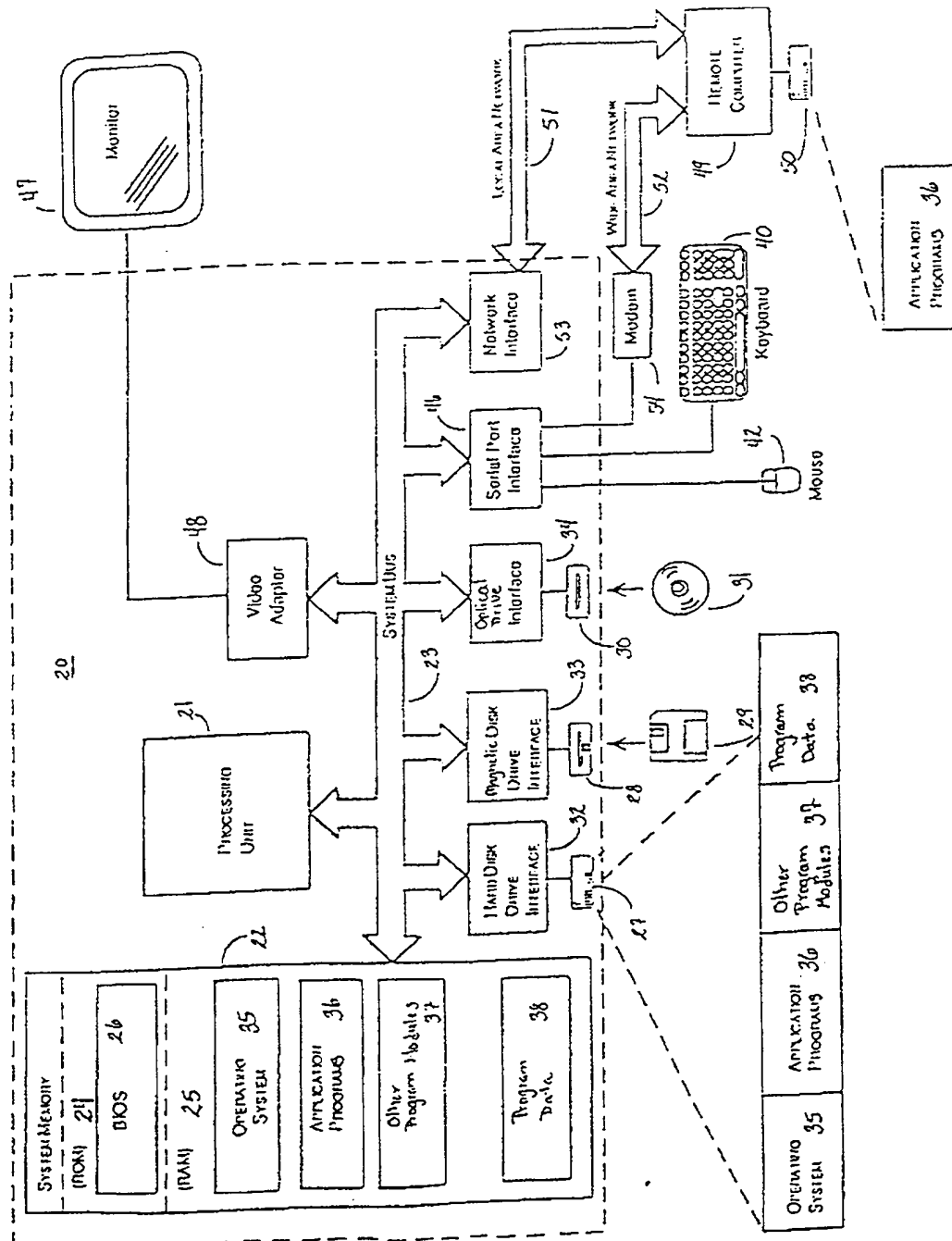


FIG. 1



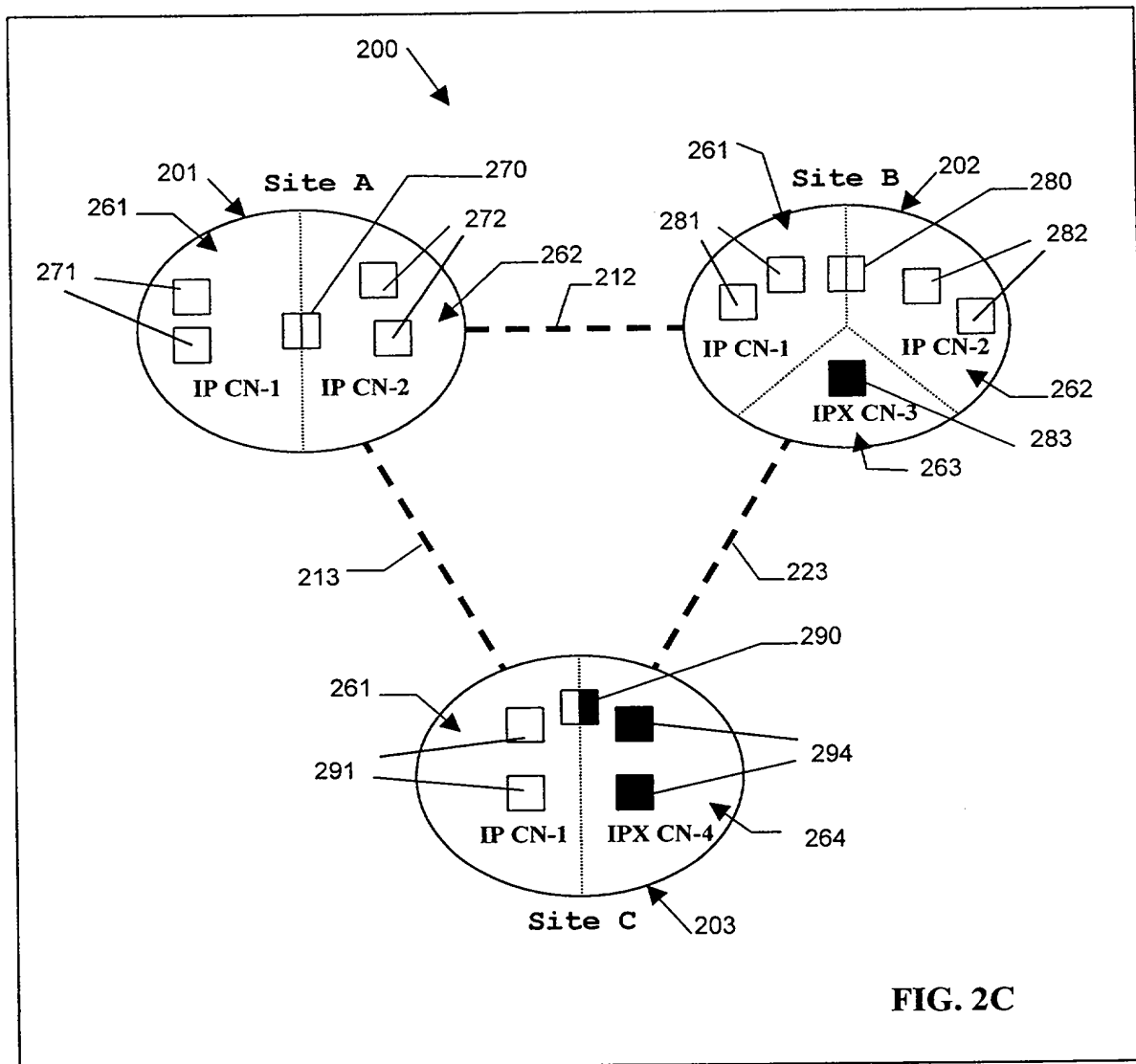
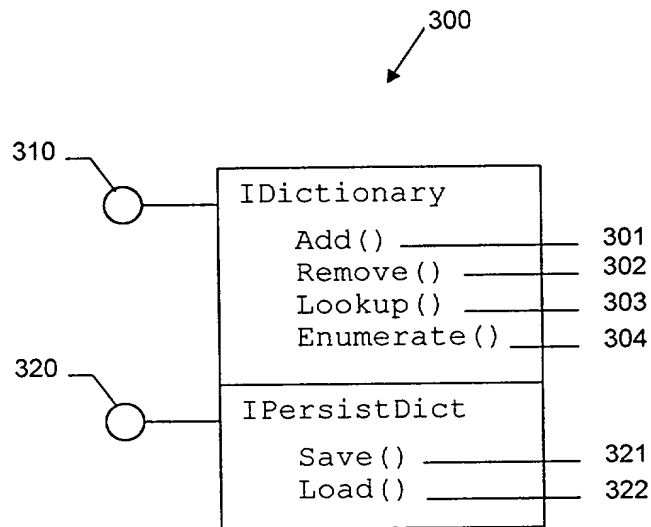
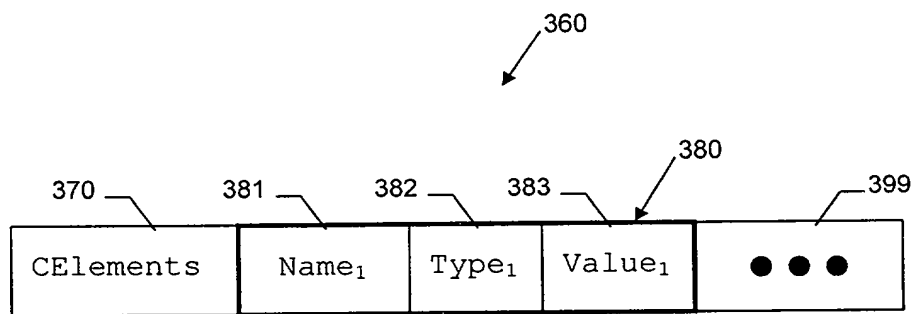


FIG. 2C

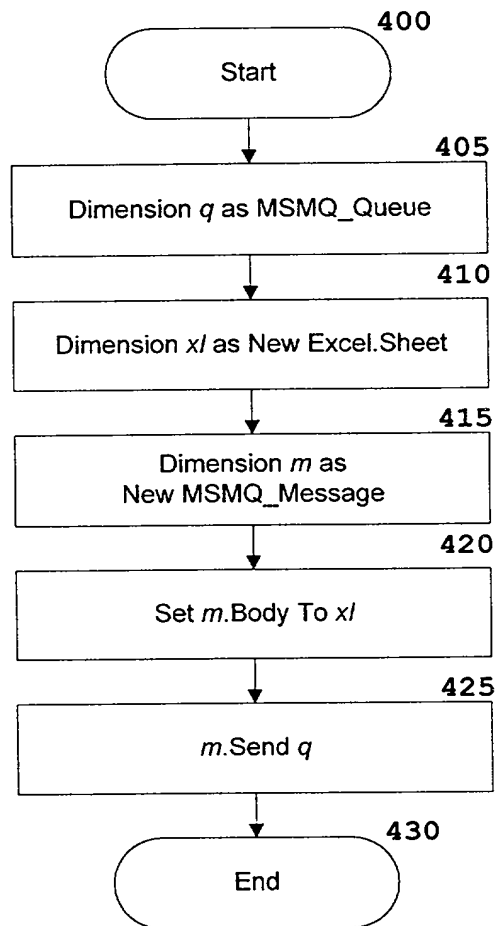
FIG. 3A - Persistent Dictionary Object



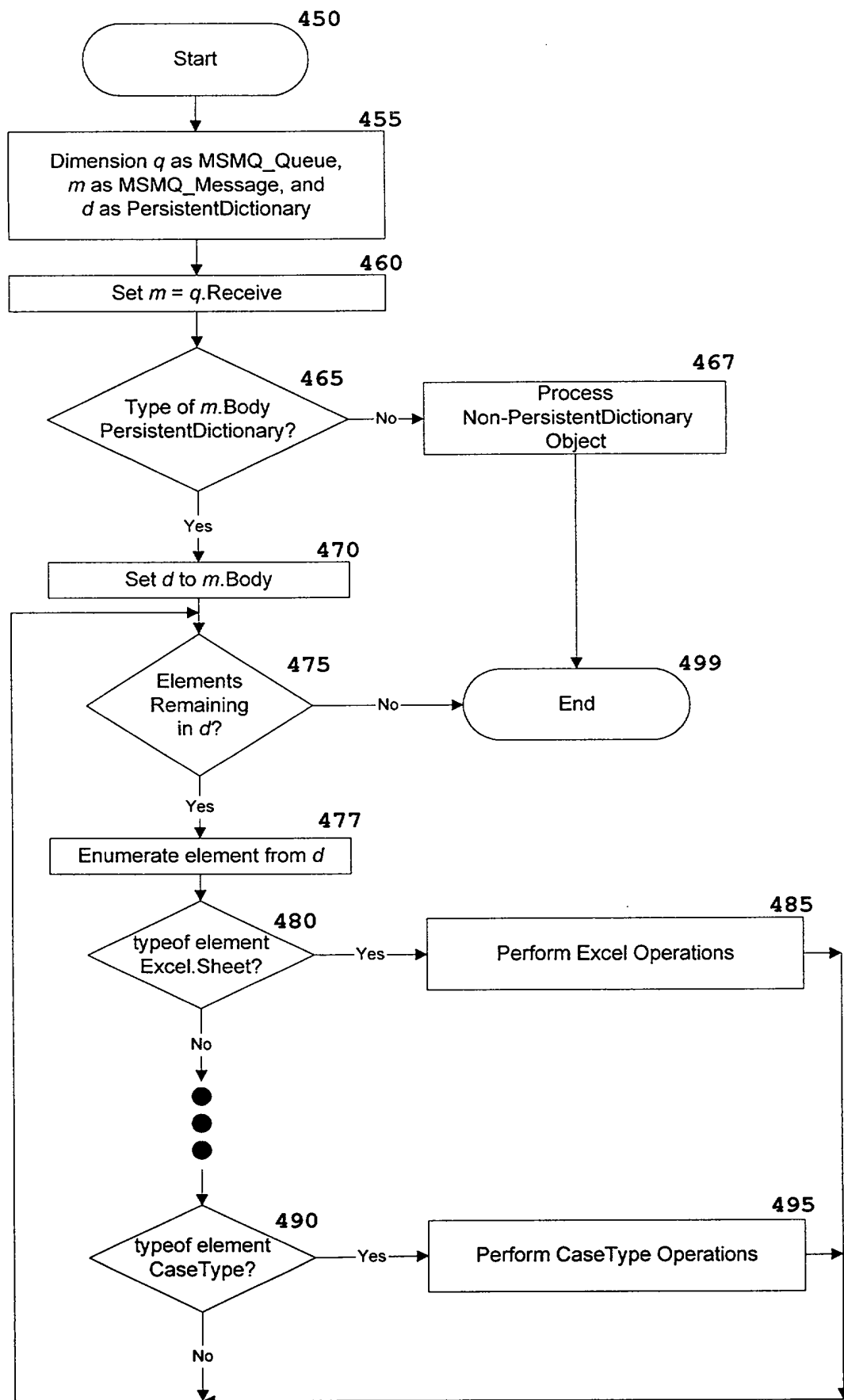
**FIG. 3A - Persistent Dictionary Object**



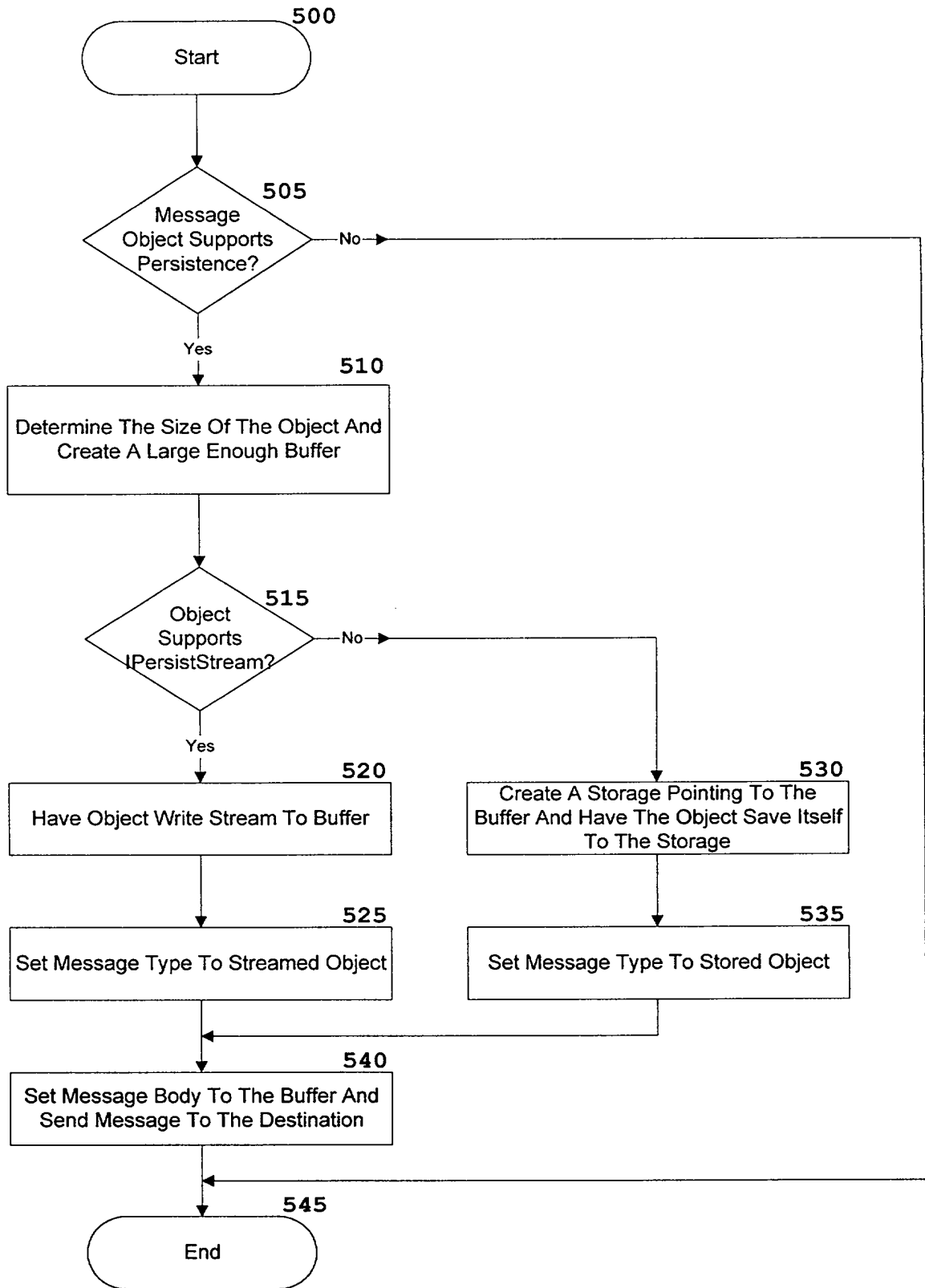
**FIG. 3B - Serialized Dictionary Object**



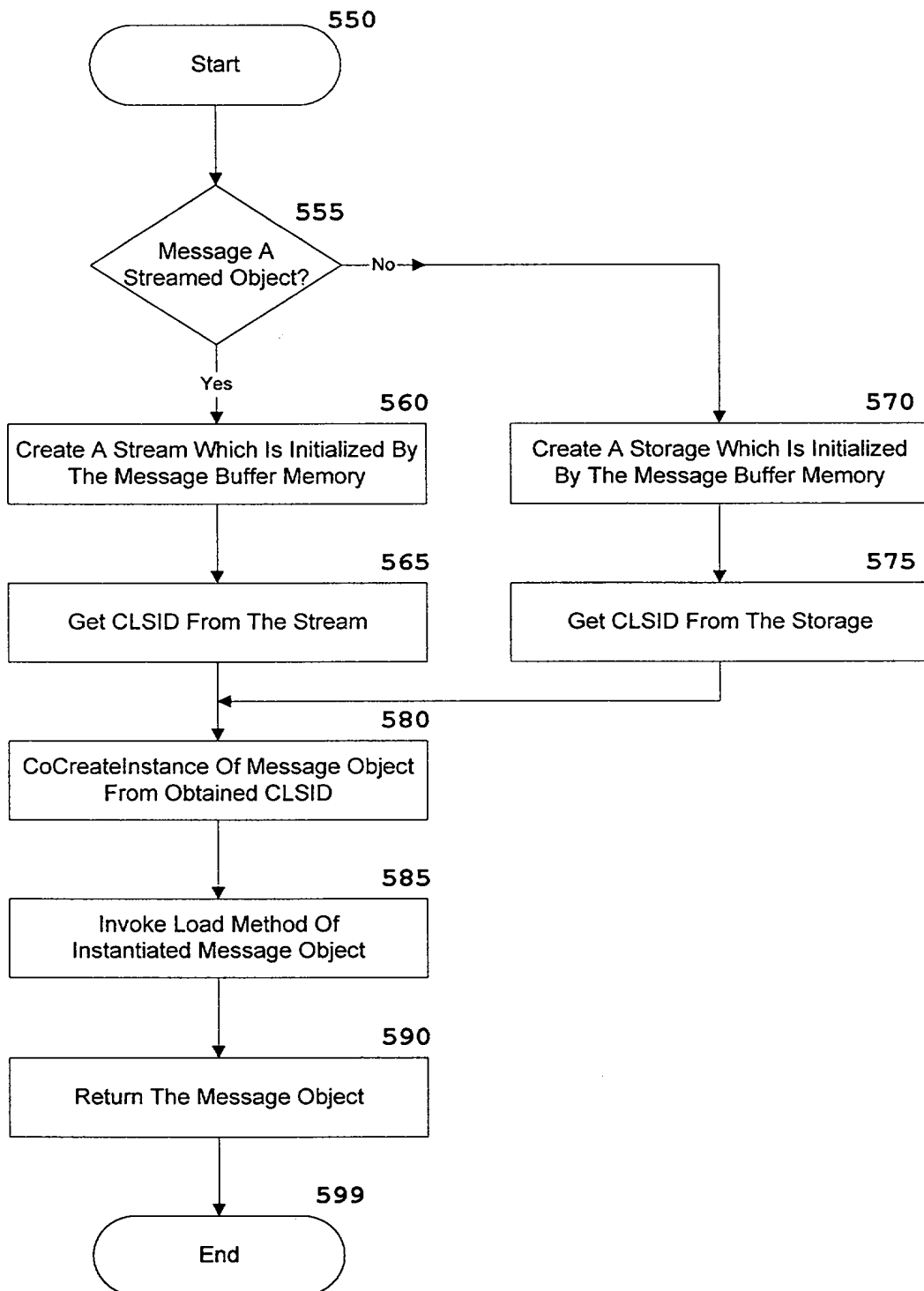
**FIG. 4A - Sending Application**



**FIG. 4B - Receiving Application**



**FIG. 5A - MSMQ Sending a Message Object**



**FIG. 5B - MSMQ Receiving a Message Object**

COMBINED DECLARATION AND POWER OF ATTORNEY

As below named inventor, I hereby declare that

This declaration is of the following type:

- ☒ original ☐ design ☐ supplemental  
☐ national stage of PCT  
☐ divisional ☐ continuation ☐ continuation-in-part

My residence, post office address, and citizenship are as stated below next to my name. I believe I am the original, first, and sole inventor (*if only one name is listed below*) or an original, first, and joint inventor (*if plural names are listed below*) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

METHOD AND APPARATUS FOR CREATING, SENDING, AND USING  
SELF-DESCRIPTIVE OBJECTS AS MESSAGES OVER A MESSAGE  
QUEUEING NETWORK

the specification of which:

- ☐ is attached hereto.  
☐ was filed on \_\_\_\_\_ as Serial No. \_\_\_\_\_ and was amended on \_\_\_\_\_ (*if applicable*).  
☒ was filed by Express Mail No. EM597736253US (*as Serial No. not known yet*).  
☐ was described and claimed in PCT International Application No. \_\_\_\_\_ filed on \_\_\_\_\_ and as amended under PCT Article 19 on \_\_\_\_\_ (*if any*).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119 of any foreign application(s) for patent or inventor's certificate or of any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed.

COUNTRY	APPLICATION	DATE OF FILING (day,month,year)	PRIORITY CLAIMED UNDER 35 USC 119			
				YES		NO
				YES		NO
				YES		NO
				YES		NO

I hereby claim the benefit pursuant to Title 35, United States Code, § 119(e) of the following United States provisional application(s):

PRIOR U.S. PROVISIONAL APPLICATIONS CLAIMING THE BENEFIT UNDER 35 USC 119(e)	
APPLICATION NO.	DATE OF FILING

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s) or PCT international application(s) designating the United States of America that is/are listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in that/those prior application(s) in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, § 1.56 which occurred between the filing date of the prior application(s) and the national or PCT international filing date of this application.

PRIOR U.S. APPLICATIONS OR PCT INTERNATIONAL APPLICATIONS DESIGNATING THE U.S. FOR BENEFIT UNDER 35 USC 120					
U.S. APPLICATIONS			Status (check one)		
U.S. APPLICATIONS	U.S. FILING DATE		PATENTED	PENDING	ABANDONED
1. 0 /					
2. 0 /					
3. 0 /					
PCT APPLICATIONS DESIGNATING THE U.S.			Status (check one)		
PCT APPLICATION NO.	PCT FILING DATE	U.S. SERIAL NOS. ASSIGNED (if any)	PATENTED	PENDING	ABANDONED
4.					
5.					
6.					

DETAILS OF FOREIGN APPLICATIONS FROM WHICH PRIORITY CLAIMED UNDER 35 USC 119 FOR ABOVE LISTED U.S./PCT APPLICATIONS				
ABOVE APPLN. NO.	COUNTRY	APPLICATION NO.	DATE OF FILING (day,month,yr)	DATE OF ISSUE (day,month,yr)
1.				
2.				
3.				
4.				
5.				
6.				

As a named inventor, I hereby appoint the following attorneys to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

Berton Scott Sheppard, Reg. 20922  
James B. Muskal, Reg. 22797  
Dennis R. Schlemmer, Reg. 24703  
Gordon R. Coons, Reg. 20821  
John E. Rosenquist, Reg. 26356  
John W. Kozak, Reg. 25117  
Charles S. Oslakovic, Reg. 27583  
Mark E. Phelps, Reg. 28461  
H. Michael Hartmann, Reg. 28423  
Bruce M. Gagala, Reg. 28844  
Charles H. Mottier, Reg. 30874  
John Kilyk, Jr., Reg. 30763  
Robert F. Green, Reg. 27555  
John B. Conklin, Reg. 30369  
James D. Zalewa, Reg. 27848  
John M. Belz, Reg. 30359

Brett A. Hesterberg, Reg. 31837  
Jeffrey A. Wyand, Reg. 29458  
Paul J. Korniczky, Reg. 32849  
Pamela J. Ruschau, Reg. 34242  
Steven P. Petersen, Reg. 32927  
John M. Augustyn, Reg. 33589  
Christopher T. Griffith, Reg. 33392  
Wesley O. Mueller, Reg. 33976  
Jeremy M. Jay, Reg. 33587  
Jeffrey B. Burgan, Reg. 35463  
Eley O. Thompson, Reg. 36035  
Mark Joy, Reg. 35562  
Allen E. Hoover, Reg. 37354  
David M. Airan, Reg. 38811  
Michael H. Tobias, Reg. 32948  
Xavier Pillai, Reg. 39799

G. Russell Thill, Reg. 39854  
Y. Kurt Chang, Reg. 41397  
Gregory C. Bays, Reg. 40505  
Carol Larcher, Reg. 35243  
Thomas A. Miller, Reg. 40091  
Gregory A. Hunt, Reg. 41085  
Patrick R. Jewik, Reg. 40456  
Thomas A. Belush, Reg. 37090  
Gary R. Jarosik, Reg. 35906  
Jeffrey J. Makeever, Reg. 37390  
Salim A. Hasan, Reg. 38175  
David J. Schodin, Reg. 41294  
Paul L. Ahern, Reg. 17020  
Theodore W. Anderson, Reg. 17035  
Noel I. Smith, Reg. 18698

I further direct that correspondence concerning this application be directed to LEYDIG, VOIT & MAYER, LTD., Two Prudential Plaza, Suite 4900, 180 North Stetson, Chicago, Illinois 60601-6780, Telephone (312) 616-5600.

I hereby declare that all statements made herein of my own knowledge are true, that all statements made on information and belief are believed to be true, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor: ILAN GABRIEL CARON

Inventor's signature \_\_\_\_\_

Date \_\_\_\_\_

Country of Citizenship: UNITED STATES

Residence: 1265 23<sup>rd</sup>. Avenue East  
Seattle, Washington 98112

Post Office Address: Same as above.